DOI 10.37882/2223-2966.2025.06.02

РАЗРАБОТКА НЕЙРОННОЙ СЕТИ ДЛЯ КЛАССИФИКАЦИИ СГЕНЕРИРОВАННЫХ ТЕКСТОВ В ОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЯХ

DEVELOPMENT OF A NEURAL NETWORK FOR CLASSIFICATION OF GENERATED TEXTS IN EDUCATIONAL INSTITUTIONS

E. Alekseeva
S. Trushin

Summary. This article discusses the features of recognition of generated texts by language models. The architectures for vector representations, as well as the architectures of recurrent networks that are most suitable for working and analyzing generated texts, are given. An open dataset from Kaggle was used to train the neural network. An algorithm based on ELMo and fully connected networks for text classification has been developed. The input data has been preprocessed. An example of the algorithm's operability is presented using the example of school essays. It is concluded that the developed neural network effectively classifies texts and provides significant support in maintaining academic integrity and will serve as a reliable tool for educational institutions.

Keywords: neural networks, artificial intelligence, deep learning, ELMo, RNN, generated text, development, algorithm, language models.

Алексеева Екатерина Сергеевна
ФГБОУ ВО Российский технологический университет — МИРЭА, г. Москва
machkt@yandex.ru
Трушин Степан Михайлович
Старший преподаватель, ФГБОУ ВО Российский технологический университет — МИРЭА, г. Москва
trushin@mirea.ru

Аннотация. В данной статье рассматриваются особенности распознавания сгенерированных текстов языковыми моделями. Приведены архитектуры для векторных представлений, а также архитектуры рекуррентных сетей, наиболее подходящих для работы и анализа сгенерированных текстов. Для обучения нейронной сети использован открытый набор данных с Kaggle. Разработан алгоритм на основании ELMo и полно связных сетей для классификации текстов. Пред обработаны входные данные. Представлен пример работоспособности алгоритма на примере школьных сочинений. Сделан вывод, что разработанная нейронная сеть эффективно классифицирует тексты и оказывает значительную поддержку в поддержании академической честности и будет служить надежным инструментом для образовательных учреждений.

Ключевые слова: нейронные сети, искусственный интеллект, глубокое обучение, ELMo, RNN, сгенерированный текст, разработка, алгоритм, языковые модели.

В эпоху цифровизации образовательного процесса вопросы академической честности приобретают особую актуальность. С появлением продвинутых языковых моделей, таких как ChatGPT, Bard, HuggingFace и т.д., возникает необходимость в разработке надежных методов для идентификации подлинности студенческих работ. В условиях цифровизации образования доступ к мощным генеративным инструментам становится всё проще и шире. Это увеличивает риск плагиата и неэтичного использования технологий, что подрывает принципы академической честности и угрожает целостности образовательного процесса.

Также, текущие системы антиплагиата, как правило, настроены на поиск прямых совпадений и перефразирований текста, что делает их недостаточно эффективными в выявлении работ, сгенерированных с помощью искусственного интеллекта (ИИ). Современные языковые модели способны создавать высококачественные тексты, которые могут быть ошибочно приняты за оригинальные работы. Это представляет собой серьёзный пробел в существующих системах обнаружения плагиата, поскольку они ещё не обладают нужными алгоритмическими решениями для адекватного анализа и классификации текстов, созданных искусственным интеллектом.

Решением вышеописанной проблемы является разработка вспомогательных методов и алгоритмов, способных точно определять, был ли текст создан студентом самостоятельно или же сгенерирован искусственным интеллектом.

Одним из наиболее подходящих методов для исследуемой задачи являются нейронные сети, поскольку они обладают способностью анализировать большие объёмы данных и выявлять сложные закономерности в текстах, а также их способность к глубокому анализу контекста и смысла текста. Нейронные сети могут обучаться на примерах оригинальных и сгенерированных текстов, способны распознавать тонкие отличия в стилистике, структуре и использовании языка, что часто невидимо для глаза человека. Кроме того, применение глубокого обучения позволяет адаптировать алгоритмы под конкретные академические требования и обновлять их

в соответствии с развитием технологий, что делает их неоценимым инструментом в поддержании академической добросовестности.

В рассматриваемой задаче особенно эффективными будут нейронные сети, предназначенные для работы с последовательными данными.

Рассмотрим наиболее популярные архитектуры нейронных сетей, специализирующиеся на работы с текстовыми данными, а именно RNN и его производные LSTM и GRU (Таблица 1).

Таблица 1.

Архитектуры RNN

Алгоритм	Краткое описание	Преимущества	Недостатки
RNN	Сеть, обрабатывающие последовательные данные с учетом предыдущего контекста благодаря внутренней памяти.	Подходят для последовательных данных, учитывают контекст, гибкие размеры ввода/вывода, способны улавливать временные зависимости в данных [1]	Проблемы с долговременной памятью, затухающие/взрывающиеся градиенты, высокая вычислительная сложность
LSTM	Разновидность RNN, предназначенная для решения проблем с долговременной памятью и проблемами, возникающими при расчете градиентов	Эффективно работает с долговременными зависимостями; избегает затухания/взрыва градиентов, способны улавливать временные зависимости в данных	Более сложная архитектура, выше вычислительные затраты по сравнению с простыми RNN
GRU	Упрощённая версия LSTM с меньшим количеством параметров, облегчающая обучение и ускоряющая вычисления	Улучшенная обработка долговременных зависимостей; меньше параметров, чем у LSTM, быстрее обучается	Может быть менее мощным в некоторых задачах по сравнению с LSTM из-за упрощённой структуры.

Кроме архитектур нейронных сетей, специализированных на классификации текстов, существуют также архитектуры, нацеленные на извлечение признаков из текста, учитывающие синтаксическую связь, морфологические особенности, семантику и другие лингвистические аспекты.

Такие архитектуры позволяют преобразовывать тексты в плотные векторные представления, отражающие их семантическое и синтаксическое значение. Эмбеддинги играют ключевую роль в понимании языка, поскольку они предоставляют информацию о словах, фразах или даже целых текстах в компактной форме [2]. Рассмотрим преимущества и недостатки широко используемых методов в этой области (Таблица 2).

Все вышеописанные подходы можно как объединять, так и использовать по отдельности. Выбор между вышеописанными архитектурами для решения задачи классификации текстов, созданных студентами или искусственными языковыми моделями, зависит от конкретной задачи, характеристик данных и доступных ресурсов организации.

Однако, наиболее эффективным подходом в исследуемой задаче окажется сочетание использования ELMo (Embeddings from Language Models) для создания контекстных эмбеддингов и применения глубоких полно связных нейронных сетей для дальнейшей обработки по следующим причинам:

- контекстуализированные эмбеддинги слов: ELMo учитывает контекстное использование слов, что обеспечивает более точное понимание их значений в различных ситуациях. Это улучшает обработку текста за счет глубокого анализа семантики слов в зависимости от контекста [4];
- высокая эффективность на средних по размеру датасетах ELMo эффективно работает с датасетами, обеспечивая глубокое понимание текста без необходимости в экстремально больших объемах тренировочных данных. Это делает его подходящим для задач, где данные ограничены, но требуется высокое качество анализа;
- гибкость и простота интеграции: полно связный слой легко интегрируется с выходами ELMo, позволяя легко адаптировать модель под специфические задачи обработки текста. Это дает свободу экспериментировать и настраивать модель для достижения наилучших результатов;
- ELMo обрабатывает тексты различного качества: благодаря глубокому пониманию контекста, ELMo успешно справляется с текстами разного стиля и качества, включая специфическую терминологию и сленг. Это важно при работе с датасетами, содержащими данные из разнообразных источников;
- относительная экономичность ресурсов с ELMo: несмотря на свою мощность, ELMo требует меньше вычислительных ресурсов по сравнению с более новыми и сложными моделями, такими как BERT. Это делает связку ELMo и полно связного слоя привлекательной для случаев, когда доступ к мощным вычислительным ресурсам ограничен, сохраняя при этом высокое качество обработки текста. [5]

Таким образом, комбинация ELMo и полно связного слоя предлагает гибкое и мощное решение для анали-

Таблица 2.

Архитектуры для векторных представлений

Алгоритм	Краткое описание	Преимущества	Недостатки
Word2Vec	Модель для генерации векторных представлений слов, обучаемая на контексте их встречаемости. Позволяет словам с похожим значением иметь близкие векторы в пространстве. [3]	Эффективность в извлечении семантических и синтаксических отношений между словами. Подходит для улучшения многих NLP задач. Требует меньше вычислительных ресурсов по сравнению с более новыми моделями.	Не учитывает контекст, в котором используется слово (одинаковые векторы для всех значений слова). Ограниченность в обработке новых или редких слов.
ELMo	Глубокая контекстуализированная модель для создания эмбеддингов слов, использующая двунаправленные LSTM сети для учета как левого, так и правого контекста слова в тексте. [3]	Генерирует динамические эмбеддинги, отражающие разные значения слова в зависимости от контекста языковых нюансов.	Требует значительных вычислительных ресурсов для обучения и использования. Может быть избыточным для некоторых более простых задач NLP.
Transformers	Инновационная архитектура, основанная на механизме внимания, позволяющая модели одновременно обрабатывать все части входных данных, в отличие от последовательной обработки в RNN и LSTM.	Обеспечивает высокую эффективность обработки параллельных данных и глубокое понимание контекста благодаря механизму внимания; позволяет моделям лучше улавливать сложные зависимости в данных.	Требует значительных вычислительных ресурсов, особенно для обучения на больших датасетах. Сложность архитектуры может увеличить время разработки и интеграции моделей.

за текстов, обеспечивая глубокое понимание контекста и семантики при относительно низких требованиях к вычислениям.

Перед тем, как рассмотреть архитектуру ELMo, для начала необходимо рассмотреть, как работают рекуррентные нейронные сети (RNN/LSTM).

Рекуррентная нейронная сеть (RNN) — это класс нейронных сетей, который эффективно обрабатывает последовательные данные (например, временные ряды или текст). В отличие от традиционных нейронных сетей, которые предполагают независимость между входами, RNN использует информацию о предыдущих состояниях для влияния на выход текущего состояния [6]. Это делает RNN особенно подходящими для задач, где контекст или порядок входных данных важны для получения корректного вывода.

Основной шаг рекуррентной нейронной сети может быть выражен следующей формулой 1:

$$h_t = \sigma\left(W \cdot x_t + U \cdot h_{t-1} + b\right), \quad (1)$$

где h_t— скрытое состояние на шаге t;

x_t — вход на шаге t;

W и U — весовые матрицы;

b — вектор смещения.

Формула 1 показывает, как на каждом шаге t текущее скрытое состояние h_t вычисляется на основе текущего входа x_t и предыдущего скрытого состояния h_{t-1}. Это позволяет RNN передавать информацию от шага к шагу во времени. Блок RNN сети представлен на Рисунке 1 [7].

Также важно уделить внимание архитектуре LSTM (Long Short-Term Memory), поскольку алгоритм ELMo, выбранный для классификации текстов, использует именно LSTM блоки для обработки текстовых данных. Это обусловлено способностью LSTM эффективно работать с длинными зависимостями в данных благодаря специализированным механизмам контроля потока информации.

LSTM управляет потоком информации через три основных компонента: входной вентиль (input gate), забывающий вентиль (forget gate) и выходной вентиль (output gate). Эти вентили определяют, какая информация будет сохранена, обновлена или удалена. Скрытое состояние h_t и состояние C_t обновляются на каждом шаге времени t с учетом текущего входа x_t и предыдущего скрытого состояния h_{t-1}:

Забывающий вентиль, определяющий какая информация из состояния ячейки будет удалена [7], представлен в формуле 2:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right), \quad (2)$$

где f_t — вектор забывающего вентиля на шаге t;

W_f — весовая матрица для забывающего вентиля t;

$[h_{t-1}, x_t]$ — конкатенация предыдущего скрытого состояния и текущего входа;

b_f — вектор смещения;

σ — сигмоидальная функция активации.

Входной вентиль, определяющий какая новая информация будет добавлена в состояние ячейки [7], представлен в формуле 3:

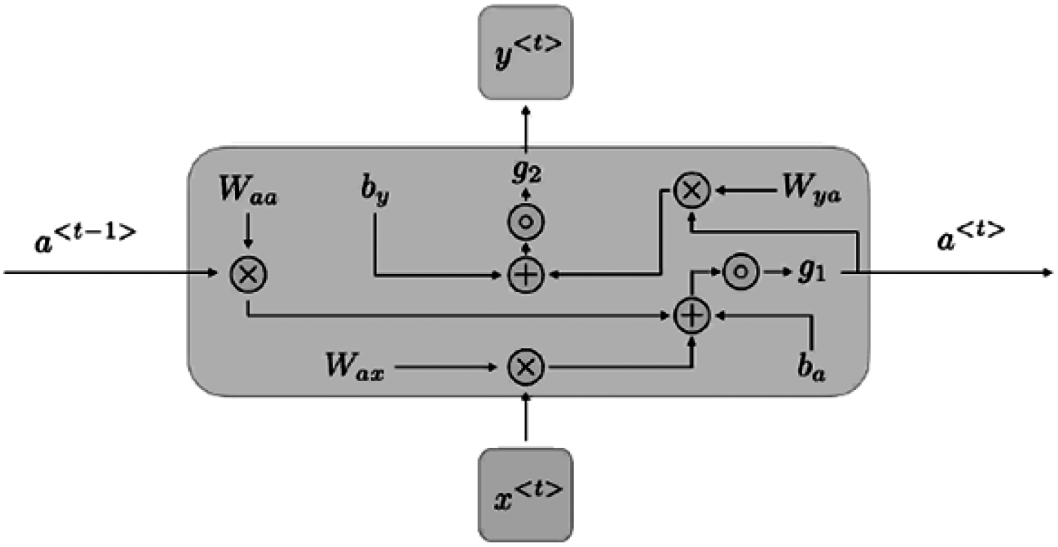

Рис. 1. Блок рекуррентной сети

$$\begin{cases} i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right), \\ \tilde{C}_t = \tanh(W_c \cdot \left[h_{t-1}, x_t\right] + b_c), \end{cases} \quad (3)$$

где i_t — вектор входного вентиля,

$\tilde{C}_t$ — вектор кандидата состояния ячейки;

W_c и W_i — весовые матрицы;

b_c и b_i — вектор смещения;

tanh — гиперболический тангенс.

Обновление состояния ячейки происходит путем умножения предыдущего состояния на забывающий вентиль (чтобы удалить ненужную информацию) и добавления новой информации [7] (формула 4):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (4)$$

где C_t — новое состояние ячейки на шаге *t*;

C_{t-1} — предыдущее состояние ячейки.

Вентиль выходного состояния. Он отвечает на вопрос о том, сколько информации из долговременной памяти следует отдавать на выход из LSTM-блока [7]. Доля вычисляется по формуле 5:

$$\begin{cases} o_t = \sigma(W_o \cdot \left[h_{t-1}, x_t\right] + b_o), \\ h_t = O_{t^*} \tanh(C_t) \end{cases} \quad (5)$$

где O_t — вектор выходного вентиля;

h_t— скрытое состояние на шаге *t*, представляет собой выход LSTM-ячейки;

W_o — весовая матрица для выходного вентиля;

b_o — вектор смещения;

tanh — гиперболический тангенс.

Наглядно работа архитектуры LSTM представлена на Рисунке 2 [8]:

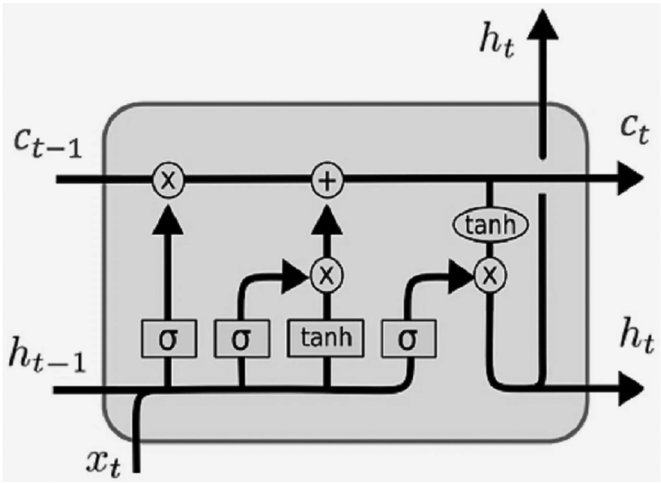

Рис. 2. Блок LSTM сети

Перейдем непосредственно к алгоритму ELMo. Алгоритм предназначен для генерации контекстуализированных представлений слов, учитывая их использование в тексте для обеспечения глубокого понимания языковых особенностей. Процесс работы алгоритма условно делится на две основные фазы: создание представлений слов на основе их контекста и их дальнейшее использование для решения конкретных задач обработки естественного языка, включая классификацию текстов. [9]

Для формирования контекстуализированных представлений слов ELMo использует двунаправленный LSTM в обучении, так что его языковая модель понимает не только следующее слово, но и предыдущее слово в предложении. Он содержит 2-слойную двунаправленную магистраль LSTM. Остаточное соединение добавляется между первым и вторым слоями. Остаточные соединения используются для того, чтобы градиенты проходили через сеть напрямую, не проходя через нелинейные функции активации. [10]

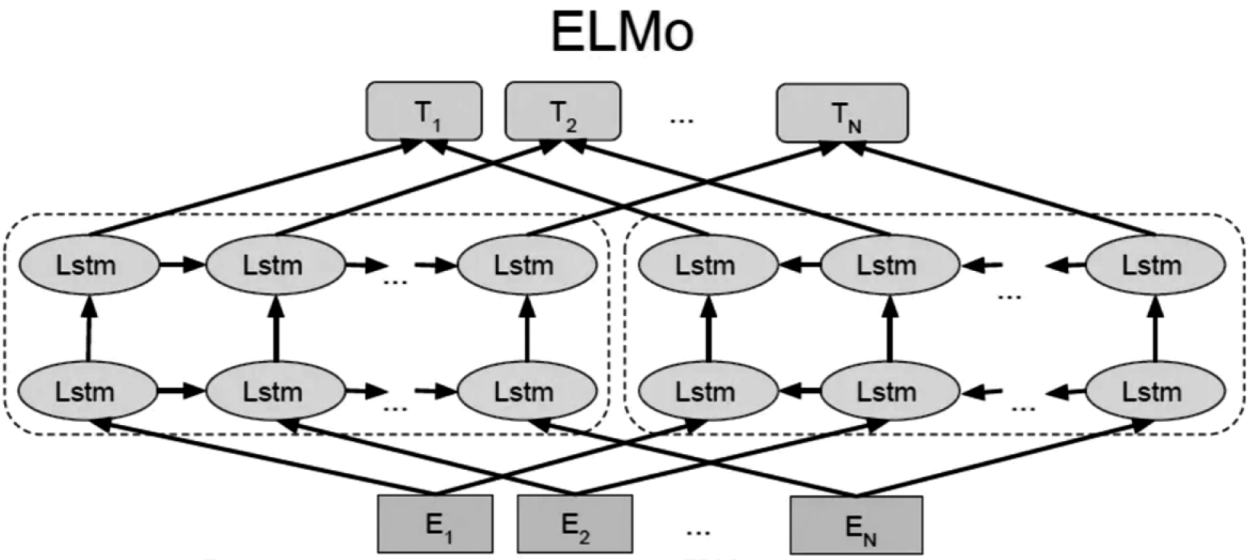

Рис. 3. Архитектура ELMo

Архитектура ELMo представлена на Рисунке 3 [11].

ELMo, аналогично большинству современных архитектур глубоких нейронных сетей, применяет метод градиентного спуска для оптимизации весов в ходе процесса обучения.

В контексте оптимизации параметров в процессе обучения, актуальностью обладает рассмотрение адаптивных методов оптимизации, среди которых выделяется AdamW. Данный метод применим к сложным архитектурам глубоких нейронных сетей, в том числе ELMo. AdamW является модификацией метода Adam, дополненной регуляризацией весов, чтобы эффект регуляризации не затухал со временем, и обобщающая способность модели была выше. [12]

Что же касаемо функции потерь, в данном случае, когда задача классификации текстов может быть сведена к определению принадлежит ли текст к одной из двух категорий (например, текст написан человеком или сгенерирован машиной), подходящим выбором является функция потерь «binary_cross_entropy_with_logits». Расчет для одного шага в процессе обучения модели может быть выражен по формуле (6):

$$l(x,y) = -\omega\left[y\log(\sigma(x)) + (1-y)\cdot\log\left(1-\sigma(x)\right)\right], \quad (6)$$

где ω — весовой коэффициент, который можно использовать для балансировки вклада разных классов в функцию потерь;

x — предсказанное значение модели для *i*-го объекта (логит);

y — истинная метка для *i*-го объекта, 0 или 1;

$\sigma(x)$ — сигмоидная функция, применяемая к логиту.

Эта функция потерь является расширением классической бинарной кросс-энтропии, которая обычно используется в задачах бинарной классификации. Особенность функции «binary_cross_entropy_with_logits» заключается в том, что она объединяет в себе сигмоидное преобразование и вычисление бинарной кросс-энтропии в одном шаге, что повышает численную стабильность процесса обучения по сравнению с использованием отдельной сигмоидной функции активации и последующего расчета кросс-энтропии. [13]

После выбора ключевых инструментов и алгоритмов, необходимо перейти к обзору и предобработке выбранного набора данных.

В контексте идентификации оригинальности текста задача нейронной сети заключается в анализе текстовых данных для выявления уникальных и повторяющихся паттернов, указывающих на потенциальное авторство студента или принадлежность текста к сгенерированным языковыми моделями. Важно на этапе предобработки определить и обработать входные данные таким образом, чтобы нейронная сеть могла эффективно извлекать и анализировать характеристики текста, включая стилистику, структуру и лексическое разнообразие, которые могут служить индикаторами его происхождения.

Данные для обучения модели играют важную роль для обучения модели машинного обучения. Они должны включать широкий спектр стилей и тем, чтобы обеспечить модель достаточным разнообразием для обучения. Важно, чтобы тексты охватывали разные области знаний и уровни сложности, что позволит системе точно классифицировать и идентифицировать особенности текстов, характерные для человеческого автора или языковой модели.

К тому же тексты должны быть актуальными и отражать реальные сценарии использования языка, что увеличит точность модели при её применении в реальных условиях.

В данной статье использовался набор данных, собранный из открытых источников и специализированных платформ, таких как Kaggle. [14]

Набор данных включает в себя многообразие сочинений, написанных школьниками старших классов и студентами на различные темы, а также сочинения, сгенерированные искусственными языковыми моделями под эти же темы. Объем собранных текстов составляет 1 Гб, что обеспечивает широкий спектр лингвистических и стилистических особенностей для анализа.

Также выбранный набор данных изначально не размечен, следовательно, придётся выделить тренировочную и обучающую выборки. При простом случайном разделении на тренировочное и тестовое множества может случиться так, что распределения тренировочного и тестового множеств окажутся не такими, как у всего исходного множества. На помощь в такой ситуации может прийти стратификация: разбиение на обучающую и тестовую выборки, сохраняющее соотношение классов, представленное в исходном наборе данных. Стратификация обеспечит сбалансированное представление разных типов текстов в обеих выборках, позволяя тем самым точно оценить эффективность модели.

Предобработка данных играет важную роль в задачах, связанных с текстами, поскольку качество и эффективность конечных результатов алгоритмов машинного обучения напрямую зависят от чистоты и структурированности входных данных.

Предобработка текстовых данных начинается с нормализации текста. Нормализация текста — это предварительный этап в NLP, который позволяет преобразовать исходный текст в такую форму, которая будет легко интерпретироваться алгоритмами машинного обучения, это включает в себя приведение всего текста к одному регистру, удаление пунктуации, лишних пробелов, табуляций и других неинформативных символов. Это необходимо для того, чтобы уменьшить шум в данных и упростить последующие этапы обработки, такие как токенизация и векторизация. [15]

Далее следует токенизация. Токенизация — процесс разбиения текста на более мелкие единицы, называемые токенами. Токены могут быть словами, фразами, символами или даже отдельными буквами. Токенизация является важным шагом, так как она определяет базовые единицы, с которыми будет работать модель машинного обучения. [16]

После токенизации может последовать лемматизация. Лемматизация — это процесс группировки различных флективных форм слов, имеющих один и тот же корень или лемму, для лучшего анализа и операций NLP. Алгоритм лемматизации удаляет аффиксы из изменяемых слов, чтобы преобразовать их в базовые слова. Стемминг же урезает слова до их корней, что иногда может быть менее точным, но более быстрым методом. Оба этих подхода помогают снизить сложность модели и ускорить процесс обучения. [17]

На Рисунках 4–5 представлены примеры исходного и пред обработанного текста.

Качественная предобработка данных не только улучшает результаты классификации текстов, но и обеспечивает более глубокое понимание контекста. А также повышает обобщающую способность нейросетевых алгоритмов.

После предобработки данных необходимо разбить выборку на три части: обучающую, валидационную и тестовую. Обучающей выборке будет положено 70 % данных, валидационной — 15 %, а на тестовую придутся оставшиеся 15 % данных.

Также нужно обратить внимание на исходный баланс классов, представленный на Рисунке 6, так как это повлияет на подход к реализации для избежания переобучения модели на определенном классе.

После оптимизации и структурирования данных, можно перейти к реализации и тестированию самого алгоритма.

Эффективное обучение модели требует организованной подачи данных по батчам, что способствует оптимизации использования памяти и ускорению процесса обучения. В этом контексте компонент DataLoader библиотеки PyTorch играет ключевую роль, автоматизируя загрузку данных и предоставляя эффективные инструменты для итерации по данным во время тренировки модели.

В основном, DataLoader работает с объектом Dataset. Чтобы использовать DataLoader, необходимо поместить данные в оболочку Dataset.

Однако, учитывая, что классы в наборе данных не сбалансированы, важно использовать сбалансированный сэмплер. Сбалансированный сэмплер гарантирует, что в каждом батче будет равное представление каждого класса, что предотвращает доминирование численно преобладающих классов и помогает улучшить обобщающую способность модели на менее представленных классах. [18]

Это было самое пляжное лето из всех, что я помню! Уже с начала июня всё говорило о том, мы с друзьями и родителями будем проводить много времени на солнце. Воздух быстро нагрелся и стало жарко. Мы часто ходили на пляж, хоть и купаться пока было рановато. Вместо этого я и мои подруги долго гуляли по берегу босыми.

Потом пришло время ненадолго расстаться — мы с родителями уезжали на Чёрное море. Уже в аэропорту я поняла, что наша «домашняя» жара — это ничто. Солнце пекло так, что было тяжело дышать. Самое то, чтобы побыть у моря. Этим мы и занялись, когда завезли вещи в отель. В июле в Сочи самый разгар сезона. На пляже было много народу, поэтому в первый день мы лишь ходили по мелководью. Зато во все последующие успевали занять место и почти не вылезали из воды.

Также за время отпуска мы несколько раз побывали в аквапарке и парке развлечений. Карусели меня не удивили, зато колесо обозрения — ещё как! Оно такое огромное — с его вершины весь Сочи как на ладони! Я даже сделала несколько памятных фотографий. Теперь они висят, приколотые к пробковой доске в моей комнате.

Многие говорят, что уезжать из путешествия грустно, но не я. Я садилась в самолёт, наполненная воспоминаниями и отдохнувшая. А ещё я знала, что дома меня ждут друзья. Остаток лето мы с ними провели вместе на городском пляже. Играли в волейбол и купались, пока вода совсем не остыла.

Это наполненное солнцем лето навсегда останется в моей памяти. Надеюсь, мы с ним ещё встретимся!

Рис. 4. Пример исходного текста

{'моря', 'отель', 'ещё', 'говорят', 'так', 'встретимся', 'дышать', 'чтобы', 'народу', 'наша', 'также', 'что', 'в', 'том', 'первый', 'все', 'но', 'фотографий', 'пляжное', 'воды', 'грустно', 'день', 'подруги', 'уезжали', 'расстаться', 'самый', 'друзья', 'много', 'аквапарке', 'по', 'его', 'даже', 'завезли', 'карусели', 'друзьями', 'помню', 'памятных', 'а', 'побыть', 'занялись', 'море', 'будем', 'удивили', 'памяти', 'стало', 'июня', 'когда', 'несколько', 'берегу', 'доске', 'наполненное', 'воздух', 'жарко', 'жара', 'времени', 'дома', 'солнце', 'вместо', 'пришло', 'поняла', 'вершины', 'весь', 'остыла', 'начала', 'у', 'занять', 'комнате', 'хоть', 'садилась', 'сочи', 'купались', 'уже', 'то', 'развлечений', 'тяжело', 'июле', 'пекло', 'во', 'всё', 'ним', 'это', 'побывали', 'играли', 'вода', 'отпуска', 'как', 'гуляли', 'ничто', 'поэтому', 'они', 'пробковой', 'и', 'сделала', 'навсегда', 'нагрелся', 'долго', 'пляж', 'всех', 'чёрное', 'быстро', 'разгар', 'о', 'говорило', 'моей', 'ждут', 'провели', 'босыми', 'потом', 'рановато', 'уезжать', 'я', 'знала', 'надеюсь', 'родителями', 'сезона', 'вещи', 'за', 'мелководью', 'мои', 'висят', 'раз', 'домашняя', 'ненадолго', 'пляже', 'не', 'с', 'такое', 'наполненная', 'самое', 'теперь', 'этого', 'обозрения', 'на', 'ними', 'место', 'волейбол', 'многие', 'останется', 'часто', 'колесо', 'вместе', 'этим', 'остаток', 'парке', 'проводить', 'время', 'оно', 'солнцем', 'из', 'аэропорту', 'последующие', 'меня', 'воспоминаниями', 'к', 'путешествия', 'лишь', 'отдохнувшая', 'совсем', 'вылезали', 'приколотые', 'лето', 'огромное', 'почти', 'ладони', 'мы', 'зато', 'пока', 'ходили', 'купаться', 'городском', 'самолёт', 'было', 'успевали'}

Рис. 5. Пример пред обработанного текста

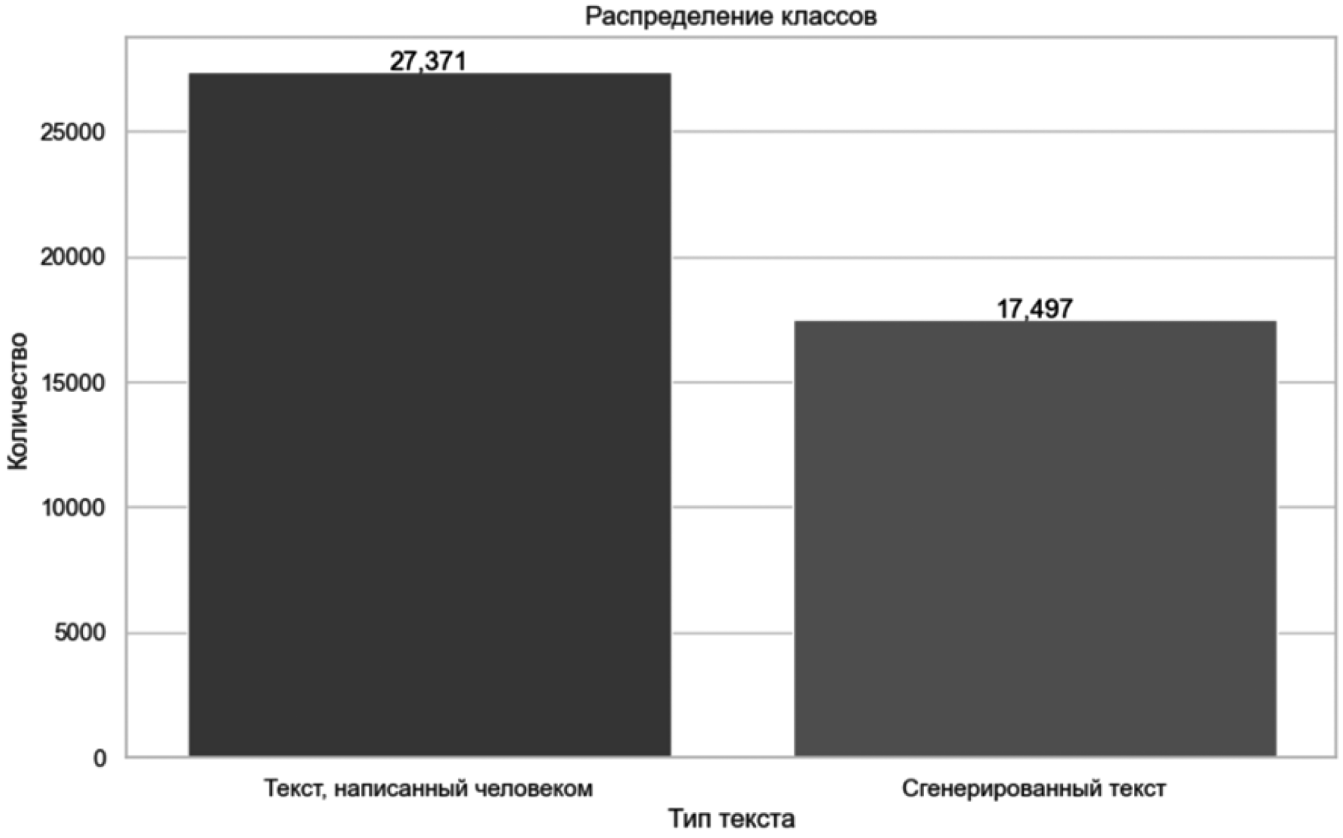

Рис. 6. Исходное распределение классов

Что касается архитектуры модели, ELMo используется в качестве основы для извлечения признаков, которые затем передаются в последующие слои нейронной сети.

В контексте текущего исследования архитектура модели дополнена полно связными слоями (FC), которые используются для дальнейшей обработки и классификации выходных данных ELMo. Эти слои позволяют формировать более сложные зависимости между представлениями слов и итоговыми классами текстов, улучшая способность модели к обобщению и точности классификации.

Такая многоуровневая архитектура модели способствует созданию мощной системы обработки текстов, способной адаптироваться к разнообразным и динамично изменяющимся условиям использования в образовательной сфере. Это, в свою очередь, обеспечивает повышение эффективности и точности в задачах классификации текстов.

Сам процесс разбивается на следующие шаги:

A. Загрузка данных: данные загружаются батчами с использованием DataLoader, который обеспечивает эффективное перемешивание и распределение данных, а также балансировку классов.
B. Прямой проход (Forward Pass): В каждом батче данные подаются в модель, которая выполняет прямой проход, вычисляя предсказания на основе текущих весов.
C. Вычисление потерь: на основе предсказаний модели и истинных значений вычисляется значение функции потерь, используя ранее определенную binary_cross_entropy_with_logits.
D. Обратное распространение (Backward Pass): вычисляется градиент функции потерь по отношению к весам модели, который затем используется для обновления весов с помощью оптимизатора.
E. Обновление параметров: оптимизатор применяет вычисленные градиенты для обновления весов модели, что должно привести к уменьшению потерь в следующих итерациях.
F. Адаптация скорости обучения: планировщик обучения может корректировать скорость обучения в зависимости от текущего шага, что помогает модели более эффективно сходиться к оптимальному решению;
G. Логирование результатов обучения: запись всех метрик в лог, а также сохранение параметров модели. [19]

Также, после каждой эпохи, производится валидация модели на отдельном наборе данных, который не ис-

пользовался во время обучения. Это позволяет оценить, как модель будет работать на незнакомых данных, и является важным шагом для предотвращения переобучения. График функции потерь и ключевой метрики во время обучения представлен на Рисунках 7–8.

После того, как нейросетевой алгоритм реализован и обучен, необходимо провести оценку на тестовом наборе данных. Это позволит оценить его обобщающую способность для новых данных.

Для оценки модели будут использовать следующие метрики: accuracy и коэффициент Gini. Каждая из упомянутых метрик будет полезна, так как в совокупности они предоставляют широкий обзор эффективности модели.

Accuracy — доля объектов, для которые правильно предсказали класс. Этот показатель может служить ориентиром для первоначального понимания эффективности модели. Однако его применение может быть не совсем практичным при работе с данными, где классы распределены неравномерно. [20]

Формально, точность (Accuracy) для задачи классификации определяется по формуле (6):

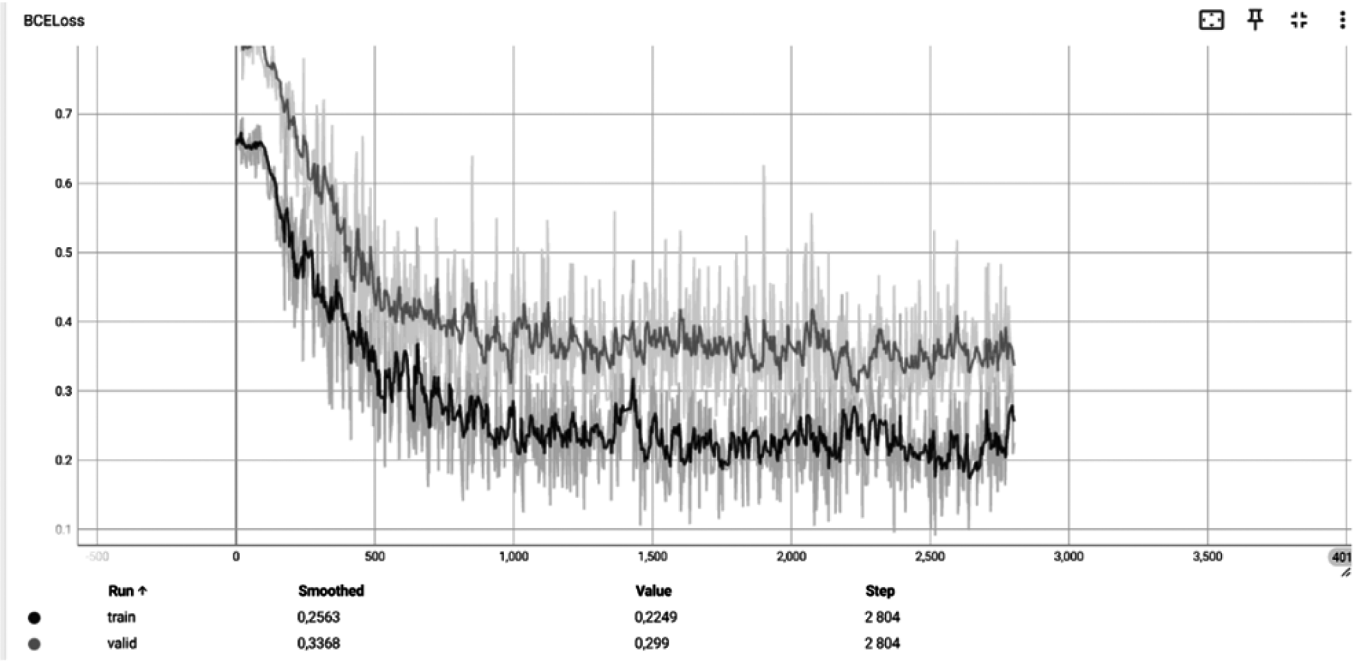

Рис. 7. График функции потерь во время обучения

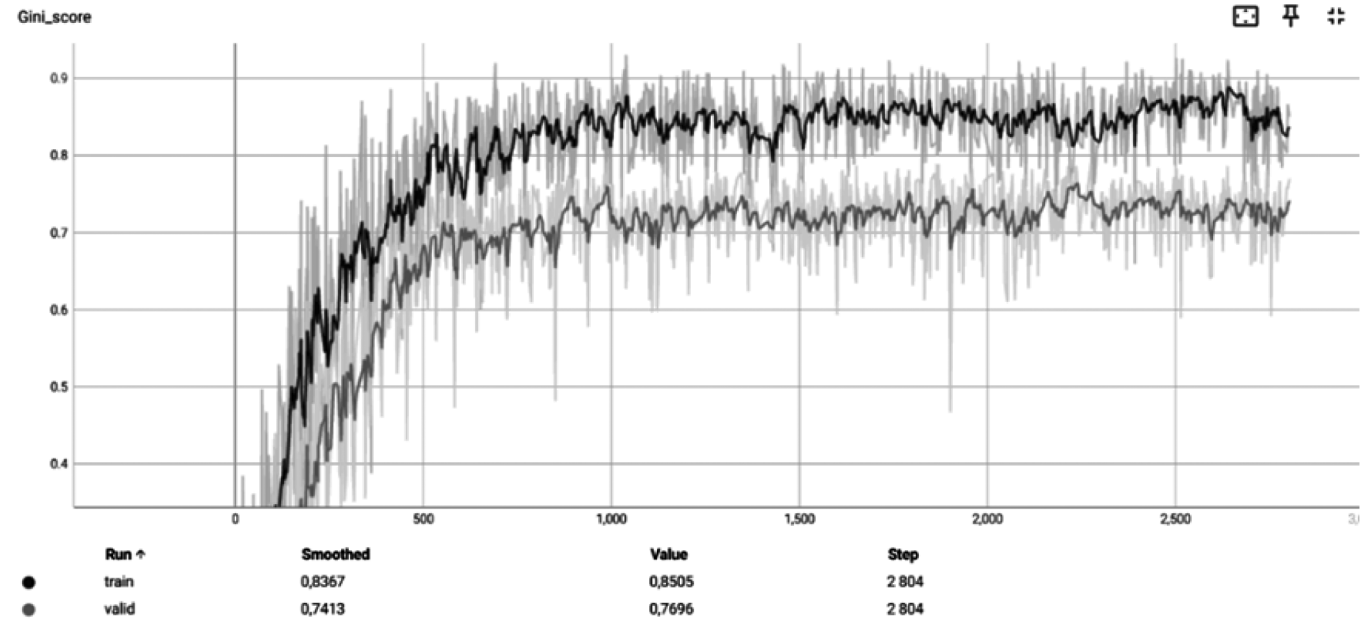

Рис. 8. График коэффициента Gini во время обучения

$$Accuracy\left(y,\hat{y}\right)=\frac{1}{N}\sum_{i=1}^{N}\mathrm{I}\left[y_i=f\left(x\right)\right], \quad (6)$$

где I — индикаторная функция, которая равна 1, если предсказанное значение $f\left(x\right)$ равно истинной метке y_i, и 0 — в противном случае;

$f\left(x\right)$ — предсказанное значение модели для *i*-го примера;

y_i — истинная метка для *i*-го примера;

N — количество всех элементов.

Коэффициент Джини (Gini coefficient) — метрика качества, которая часто используется при оценке предсказательных моделей в задачах бинарной классификации в условиях несбалансированности классов целевой переменной. Коэффициент Джини применяется для количественной оценки разделения классов, где значение 0 соответствует отсутствию разделения, а 1 — идеальному разделению. [21]

Коэффициент Джини (Gini coefficient) определяется по формуле (7):

$$Gini=2\times AUC-1 \quad (7)$$

где AUC — это площадь под кривой ошибок классификации, т. е. доля пар объектов вида (объект класса 1, объект класса 0), которые алгоритм верно упорядочил.

В ходе оценки модели на тестовой выборке были получены результаты, отраженные в Таблице 3.

Таблица 3.

Результаты оценки модели

Метрика	Значение
Gini coefficient	0.81
Accuracy	0.88

Из результатов видно, что значение Коэффициент Джини (Gini coefficient) равен 0.81. Это означает очень хорошее разделение классов, что делает модель достаточно надежной в плане идентификации принадлежности объектов к одному из двух классов.

Значение accuracy говорит нам о том, что в 88 % случаев мы правильно идентифицируем класс объекта. Такой уровень точности может свидетельствовать о хорошем качестве модели.

Также на Рисунках 9–10 представлен пример работоспособности модели на двух коротких примерах: сгенерированном сочинении и написанной человеком.

На первом примере проверим текст, сгенерированный с помощью языковой модели ChatGPT.

```
if __name__ == "__main__":
    # Загрузка модели
    elmo = ELMo.load_model("logs/2024-05-21_14-08-59")
    text = input("Введите текст: ")

    # Преобразование текста в токены
    inputs_token = dataset.to_tokens_ids(text)

    out_probabilities = elmo.model(inputs_token)

    print(f"Вероятность: {out_probabilities:.4f}")
```

Executed at 2024.05.21 18:43:57 in 3s 522ms

```
Модель загружена: logs/2024-05-21_14-08-59
Вероятность: 0.7763
```

Рис. 9. Проверка сгенерированного текста

В данном примере, вероятность того, что текст сгенерирован с помощью ИИ составляет 77.6 %.

Далее проверим текст, написанный школьником в качестве сочинения.

```
if __name__ == "__main__":
    # Загрузка модели
    elmo = ELMo.load_model("logs/2024-05-21_14-08-59")
    text = input("Введите текст: ")

    # Преобразование текста в токены
    inputs_token = dataset.to_tokens_ids(text)

    out_probabilities = elmo.model(inputs_token)

    print(f"Вероятность: {out_probabilities:.4f}")
```

Executed at 2024.05.21 18:43:33 in 3s 537ms

```
Модель загружена: logs/2024-05-21_14-08-59
Вероятность: 0.2429
```

Рис. 10. Проверка текста, написанного человеком

Здесь же, вероятность того, что текст сгенерирован с помощью ИИ составляет всего 24.2 %.

Подытожив все результаты метрик и тестирования, можно сделать вывод, что разработанная модель нейронной сети эффективно классифицирует тексты, созданные студентами от текстов созданными языковыми моделями, что позволяет её легко интегрировать в образовательную информационную систему. Разработанная модель нейронной сети окажет значительную поддержку в поддержании академической честности и будет служить надежным инструментом для образовательных учреждений.

ЛИТЕРАТУРА

1. FasterCapital [Электронный ресурс]. — Нейронные сети: как использовать нейронные сети для прогнозирования инвестиций. — Режим доступа: https://clck.ru/3BNc5v
2. Спивак А.И., Лапшин С.В., Лебедев И.С. Классификация коротких сообщений с использованием векторизации на основе ELMO // Известия Тульского государственного университета. Технические науки. — 2019. — № 10. — С. 410–418.
3. Боровков Н.А., Фураев Ф.И. Использование языковой модели ELMO для задачи аспектно-ориентированного анализа тональности // Сборник докладов Санкт-Петербургского государственного университета аэрокосмического приборостроения (СПб.). — 2019. — Том, Часть I. — С. 301–305.
4. QUData [Электронный ресурс]. — ML: Буквенный и семантический эмбединг. — Режим доступа: https://qudata.com/ml/ru/NN_Embedding_Elmo.html
5. Medium [Электронный ресурс]. — NLP Step-by-Step Guide to Learning ELMo to Extract Functions from Text. — Режим доступа: https://medium.com/analytics-vidhya/a-step-by-step-nlp-guide-to-learn-elmo-for-extracting-features-from-text
6. Сухарев А.Л., Жалнин Р.В., Федосин С.А. Разработка и реализация автоматизированного мордовско-русского переводчика на основе искусственных нейронных сетей LSTM-RNN // Научно-технический вестник Поволжья. — 2019. — № 6. — С. 126–128.
7. Гранкина Д.М., Макаров К.С. Алгоритм LSTM для прогнозирования будущих трат с использованием данных временных рядов // Сборник научных статей по материалам III Всероссийской конференции, Курск. — 2022. — С. 109–121.
8. Zheng Y., Han L., Yu J., Yu R. Driving risk assessment under the connected vehicle environment: a CNN-LSTM modeling approach // Digital Transportation and Safety. — 2023. — № 2(3). — С. 211–219.
9. Skiba A., Batura T. Comparison of ELMO-based models on the named entity recognition task // Bulletin of the Novosibirsk Computing Center. Series: Computer Science. — 2023. — № 47. — С. 33–41.
10. Medium [Электронный ресурс]. — Learn how to build powerful contextual word embeddings with ELMo. — Режим доступа: https://medium.com/saarthi-ai/elmo-for-contextual-word-embedding-for-text-classification-24c9693b0045.
11. Русские Блоги [Электронный ресурс]. — Понимание BERT: исчерпывающее руководство по революционной структуре НЛП. — Режим доступа: https://russianblogs.com/article/58971605085/
12. Монтик Н.С. Сравнение различных оптимизаторов на датасете MNIST и его вариациях при использовании функции LeakyReLU // Сборник статей Международной научно-практической конференции. — 2024. — С. 182–187.
13. PyTorch [Электронный ресурс]. — BCEWithLogitsLoss. — Режим доступа: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html
14. Kaggle [Электронный ресурс]. — Режим доступа: https://www.kaggle.com/
15. Фомин В.В., Фомина И.К. Исследование эффективности части-речевой нормализации в NLP-алгоритмах классификации русскоязычных текстов // Актуальные проблемы экономики и управления. — 2022. — № 3(36). — С. 87–91.
16. Your Todo [Электронный ресурс]. — Обработка текстов в нейронных сетях. — Режим доступа: https://www.yourtodo.ru/posts/obrabotka-tekstov-v-nejronnyih-setyah/
17. Жердева М.В., Артюшенко В.М. Стемминг и лемматизация в Lucene.NET // Вестник Московского государственного университета леса — Лесной вестник. — 2016. — Т. 20. — № 3. — С. 131–134.
18. Татаренков В.С. Программная реализация собственного набора данных лиц пользователей для выполнения обучения нейронной сети в системе идентификации и аутентификации по лицу с помощью инструментов библиотеки PyTorch // Сборник избранных статей Всероссийской (национальной) научной конференции, Санкт-Петербург. — 2022. — С. 28–31.
19. Habr [Электронный ресурс]. — Продвинутое использование библиотеки PyTorch: от подготовки данных до визуализации. — Режим доступа: https://habr.com/ru/articles/553716/
20. Гурина А.О., Елисеев В.Л. Критерий оценки качества классификации за пределами обучающей выборки // Вестник Московского энергетического института. — 2022. — № 1. — С. 98–110.
21. Turkoglu M.O., D'Aronco S., Wegner J.D., Schindler K. Gating Revisited: Deep Multi-Layer RNNs That Can be Trained // ICLR 2020 Conference Withdrawn Submission. — 2019.

Журнал «Современная наука: актуальные проблемы теории и практики»